Final Group Project Report - Database Design and Implementation for a Bicycle Store

Sebastian Gonzalez Uribe

Juan José Vargas Gomez

Mario Fernando Zamudio Portilla

University of Niagara Falls

CPSC 500-1: Sql Databases

Abbas Hamze

December 9, 2024

Introduction

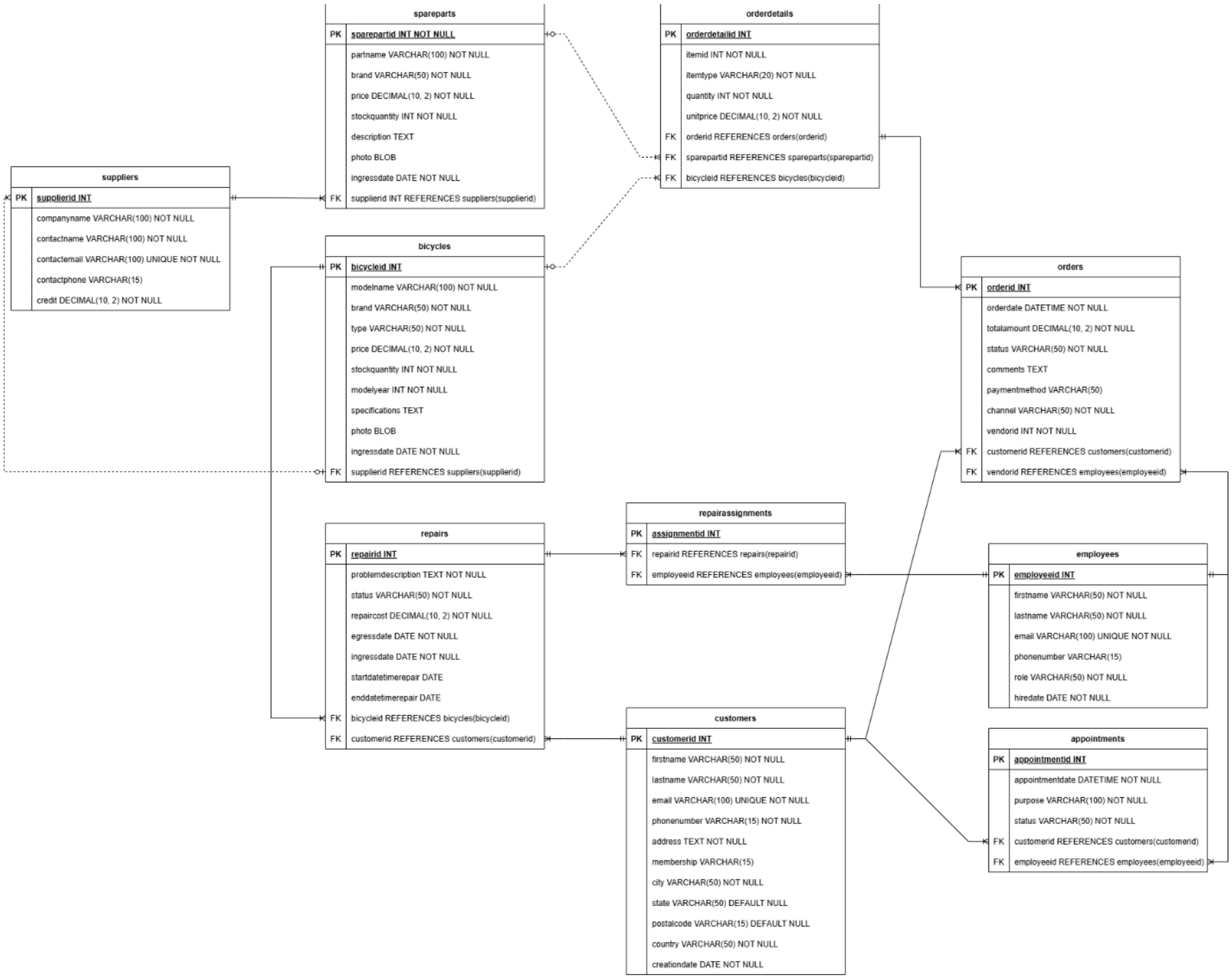
Nestled in the heart of downtown Niagara Falls, a delightful bicycle shop offers a wide array of bicycles for sale, along with expert repair services. This quaint establishment is dedicated to fostering a passion for cycling in the community and is eager to enhance its operations. To achieve this, the shop seeks to increase its sales figures while also optimizing the management of its repair schedules. To address these goals effectively, we propose a detailed and comprehensive database design that will meticulously store and organize all pertinent information. This system will significantly streamline the processes of tracking sales and scheduling repairs, ultimately enhancing overall efficiency and customer satisfaction.

Database Design

This proposal is oriented to manage the principal actors into the process to manage the sales and the repairs for the bicycle customers in the store and online. Once, the requirement and the business ideas were reviewed, the next scheme is proposed to feed and maintain the information for the company:

Figure 1

Entities Relationship Diagram


spareparts
PK sparepartid INT NOT NULL
partname VARCHAR(100) NOT NULL
brand VARCHAR(50) NOT NULL
price DECIMAL(10, 2) NOT NULL
stockquantity INT NOT NULL
description TEXT
photo BLOB
ingressdate DATE NOT NULL
FK supplierid INT REFERENCES suppliers(supplierid)
orderdetails
PK orderdetailid INT
itemid INT NOT NULL
itemtype VARCHAR(20) NOT NULL
quantity INT NOT NULL
unitprice DECIMAL(10, 2) NOT NULL
FK orderid REFERENCES orders(orderid)
FK sparepartid REFERENCES spareparts(sparepartid)
FK bicycleid REFERENCES bicycles(bicycleid)
suppliers
PK supplierid INT
companyname VARCHAR(100) NOT NULL
contactname VARCHAR(100) NOT NULL
contactemail VARCHAR(100) UNIQUE NOT NULL
contactphone VARCHAR(15)
credit DECIMAL(10, 2) NOT NULL
bicycles
PK bicycleid INT
modelname VARCHAR(100) NOT NULL
brand VARCHAR(50) NOT NULL
type VARCHAR(50) NOT NULL
price DECIMAL(10, 2) NOT NULL
stockquantity INT NOT NULL
modelyear INT NOT NULL
specifications TEXT
photo BLOB
ingressdate DATE NOT NULL
FK supplierid REFERENCES suppliers(supplierid)
orders
PK orderid INT
orderdate DATETIME NOT NULL
totalamount DECIMAL(10, 2) NOT NULL
status VARCHAR(50) NOT NULL
comments TEXT
paymentmethod VARCHAR(50)
channel VARCHAR(50) NOT NULL
vendorid INT NOT NULL
FK customerid REFERENCES customers(customerid)
FK vendorid REFERENCES employees(employeeid)
repairassignments
PK assignmentid INT
FK repairid REFERENCES repairs(repairid)
FK employeeid REFERENCES employees(employeeid)
repairs
PK repairid INT
problemdescription TEXT NOT NULL
status VARCHAR(50) NOT NULL
repaircost DECIMAL(10, 2) NOT NULL
egressdate DATE NOT NULL
ingressdate DATE NOT NULL
startdatetimerepair DATE
enddatetimerepair DATE
FK bicycleid REFERENCES bicycles(bicycleid)
FK customerid REFERENCES customers(customerid)
employees
PK employeeid INT
firstname VARCHAR(50) NOT NULL
lastname VARCHAR(50) NOT NULL
email VARCHAR(100) UNIQUE NOT NULL
phonenumber VARCHAR(15)
role VARCHAR(50) NOT NULL
hiredate DATE NOT NULL
customers
PK customerid INT
firstname VARCHAR(50) NOT NULL
lastname VARCHAR(50) NOT NULL
email VARCHAR(100) UNIQUE NOT NULL
phonenumber VARCHAR(15) NOT NULL
address TEXT NOT NULL
membership VARCHAR(15)
city VARCHAR(50) NOT NULL
state VARCHAR(50) DEFAULT NULL
postalcode VARCHAR(15) DEFAULT NULL
country VARCHAR(50) NOT NULL
creationdate DATE NOT NULL
appointments
PK appointmentid INT
appointmentdate DATETIME NOT NULL
purpose VARCHAR(100) NOT NULL
status VARCHAR(50) NOT NULL
FK customerid REFERENCES customers(customerid)
FK employeeid REFERENCES employees(employeeid)

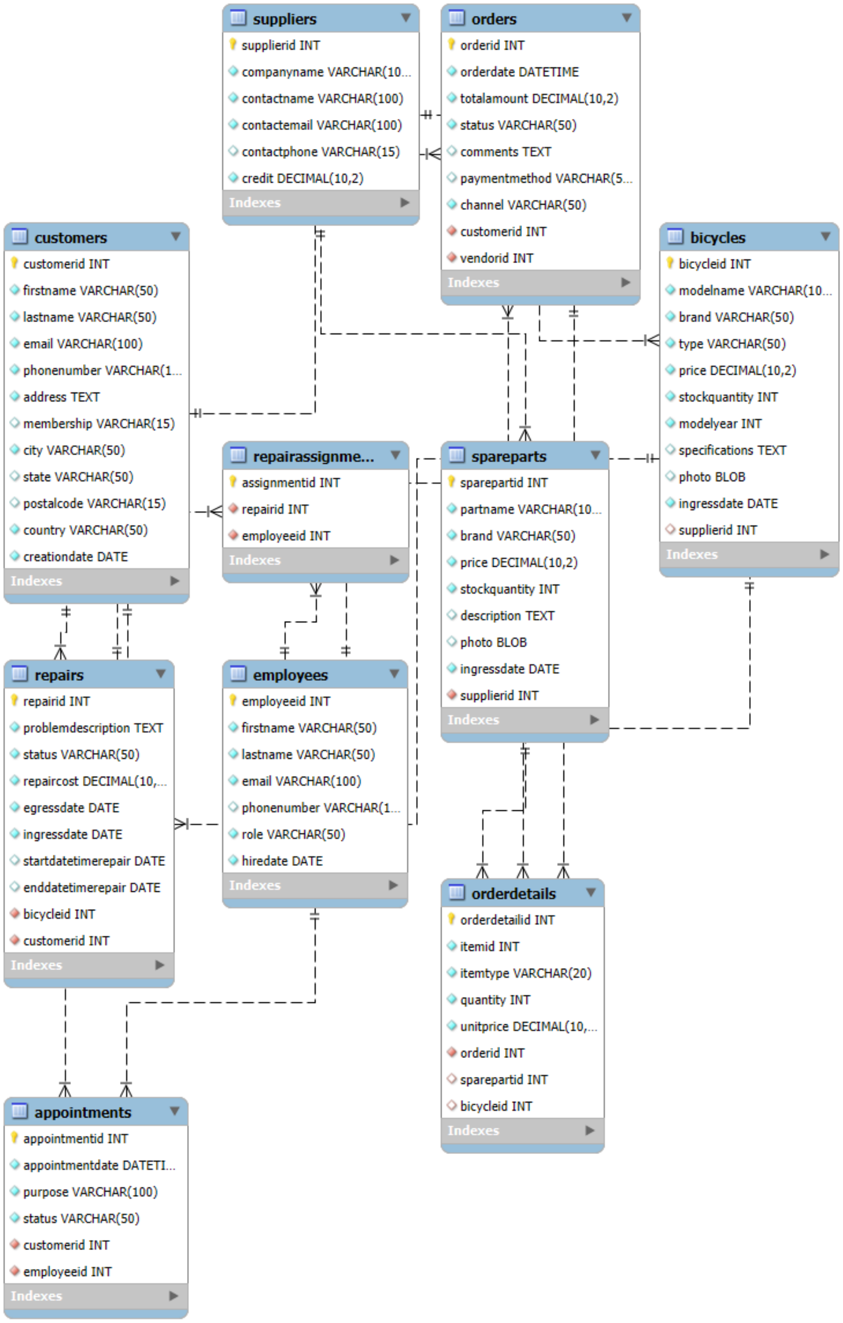
This Diagram could be accessed through the following link:

https://drive.google.com/file/d/1Awfz7Ma47DM-SGKxfq7LjkM-YMPyq_tQ/view?usp=sharing

Additionally, when the diagram was created into the MySQL database, the next diagram was generated with the objects created into the database:

Figure 2

Workbech Entity Relationship Diagram



Entities:

Table 1

Customers: Table to handle customer information

Attribute	*Type*	*Description*
customerid	INT	PRIMARY KEY
firstname	VARCHAR(50)	Firs name of the customer
lastname	VARCHAR(50)	Last name of the customer
email	VARCHAR(100)	unique and format expected xxx@yyy.zz
phonenumber	VARCHAR(15)	format expected +xx-yyyyyyyyyy
address	TEXT	Principal address of the customer
membership	VARCHAR(15)	'No','Basic','Premium'
city	VARCHAR(50)	City of the customer
state	VARCHAR(50)	State of the city
postalcode	VACHAR(15)	Postal code of the customer address
country	VARCHAR(50)	DEFAULT 'Canada'
creationdate	DATE	

Table 2

Suppliers: Table to handle supplier information.

Attribute	*Type*	*Description*
supplierid	INT	PRIMARY KEY
companyname	VARCHAR(100)	Name of the supplier company
contactname	VARCHAR(100)	Name of the contact in the supplier company

contactemail	VARCHAR(100)	Email of the contact in the supplier company, unique and format expected xxx@yyy.zz
contactphone	VARCHAR(15)	Phone number of the supplier contact, format expected +xxyyyyyyyyyy
credit	DECIMAL(10, 2)	Amount maximum of the credit allowed from the supplier

Table 3

Employees: Table to handle employee information.

Attribute	*Type*	*Description*
employeeid	INT	PRIMARY KEY
firstname	VARCHAR(50)	
lastname	VARCHAR(50)	
email	VARCHAR(100)	unique and format expected xxx@yyy.zz
phonenumber	VARCHAR(15)	format expected +xx-yyyyyyyyyy
role	VARCHAR(50)	
hiredate	DATE	

Table 4

Bicycles: Table to store the bicycles available and their basic information.

Attribute	*Type*	*Description*
bicycleid	INT	PRIMARY KEY
modelname	VARCHAR(100)	Name of the bicycle model
brand	VARCHAR(50)	Make of the bicycle model

type	VARCHAR(50)	Type of the bicycle
price	DECIMAL(10, 2)	Price to sell
stockquantity	INT	Number of units available in inventory
modelyear	INT	Year of the bicycle model, format YYYY (1900-2050)
specifications	TEXT	Description or additional characteristics of the bicycle
photo	BLOB	Real image of the bicycle
ingressdate	DATE	Date when the units were registered in the inventory
supplierid	INT	FOREIGN KEY REFERENCES bicycle_store.suppliers(supplierid)

Table 5

Spareparts: Table to store the spare parts available and their basic information.

Attribute	*Type*	*Description*
sparepartid	INT	PRIMARY KEY
partname	VARCHAR(100)	Name of the spare
brand	VARCHAR(50)	Make of the spare
price	DECIMAL(10, 2)	Price for unit
stockquantity	INT	Number of units available in inventory
description	TEXT	Description of the spare
photo	BLOB	Real image of the spare
ingressdate	DATE	Date when the units were registered in the inventory
supplierid	INT	FOREIGN KEY REFERENCES bicycle_store.suppliers(supplierid)

Table 6

Appointments: Table to manage the appointment for repairs.

Attribute	*Type*	*Description*
appointmentid	INT	PRIMARY KEY
appointmentdate	DATETIME	Date to schedule the appointment in the store
purpose	VARCHAR(100)	Reason to the appointment
status	VARCHAR(50)	Status of the appoinment 'Scheduled','Confirmed','Cancelled', 'Finished'
customerid	INT	FOREIGN KEY REFERENCES bicycle_store.customers(customerid)
employeeid	INT	FOREIGN KEY REFERENCES bicycle_store.employees(employeeid)

Table 7

Orders: Table to store the order information

Attribute	*Type*	*Description*
orderid	INT	PRIMARY KEY
orderdate	DATETIME	Date when the order was placed
totalamount	DECIMAL(10, 2)	Total amount in CAD of the order
status	VARCHAR(50)	'Cancelled','Payed','Pending'
comments	TEXT	Description text of the order
paymentmethod	VARCHAR(50)	Method used to receive the payment from the customer

channel	VARCHAR(50)	Commercial channel where the order was received ,'Store','App','WebPage', 'Partner'
customerid	INT	FOREIGN KEY REFERENCES bicycle_store.customers(customerid)
vendorid	INT	FOREIGN KEY REFERENCES bicycle_store.employees(employeeid)

Table 8

Orderdetails: Table to store the order details

Attribute	*Type*	*Description*
orderdetailid	INT	PRIMARY KEY
itemid	INT	Identifier of the item included
itemtype	VARCHAR(20)	Type of item in detail, 'Bicycle' or 'SparePart' or 'Repair'
quantity	INT	Number of items for this detail
unitprice	DECIMAL(10, 2)	Price by item or unit
orderid	INT	FOREIGN KEY REFERENCES bicycle_store.orders(orderid)
sparepartid	INT	FOREIGN KEY REFERENCES bicycle_store.spareparts(sparepartid)
bicycleid	INT	FOREIGN KEY REFERENCES bicycle_store.bicycles(bicycleid)

Table 9

Repairs: Table to store the repair information.

Attribute	*Type*	*Description*
repairid	INT	PRIMARY KEY
problemdescription	TEXT	Brief description of the problem reported by the customer
status	VARCHAR(50)	Status where the repair is on going, 'Cancelled','Ongoing','Finished'
repaircost	DECIMAL(10, 2)	Cost estimated for the repair or revision
egressdate	DATE	Date when the bicycle was returned to the client
ingressdate	DATE	Date when the bicycle was received from the client
startdatetimerepair	DATE	Date when the bicycle started to receive the service from the technician
enddatetimerepair	DATE	Date when the bicycle was served by the technician, this value must be mayor of the startdatetimerepair
bicycleid	INT	FOREIGN KEY REFERENCES bicycle_store.bicycles(bicycleid)
customerid	INT	FOREIGN KEY REFERENCES bicycle_store.customers(customerid)

Table 10

Repairassignments: Table to store the details of the repair assignment with the technician.

Attribute	*Type*	*Description*
assignmentid	INT	PRIMARY KEY

repairid	INT	FOREIGN KEY REFERENCES bicycle_store.repairs(repairid)
employeeid	INT	FOREIGN KEY REFERENCES bicycle_store.employees(employeeid)

Data Definition Statements (DDL)

Based on the previous guidelines we generate the next queries to create our tables in MySQL.

```
CREATE SCHEMA bicycle_store ;

USE bicycle_store;


-- 1. customers

CREATE TABLE bicycle_store.customers (

        customerid INT PRIMARY KEY,

        firstname VARCHAR(50) NOT NULL,

        lastname VARCHAR(50) NOT NULL,

        email VARCHAR(100) UNIQUE  NOT NULL, -- unique and format expected xxx@yyy.zz

        phonenumber VARCHAR(15) NOT NULL, -- format expected +xx-yyyyyyyyy

        address TEXT NOT NULL,

        membership VARCHAR(15),   -- 'No','Basic','Premium'

        city VARCHAR(50) NOT NULL,

        state VARCHAR(50),

        postalcode VARCHAR(15),

        country VARCHAR(50) NOT NULL DEFAULT 'Canada',

        creationdate DATE NOT NULL DEFAULT (CURRENT_DATE),

        CONSTRAINT  chk_membership_cust CHECK (membership IN ('No','Basic','Premium')),
            CONSTRAINT  chk_mail_cust CHECK (email REGEXP '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'),
            CONSTRAINT  chk_phonenumber_cust CHECK (phonenumber REGEXP '^\+[0-9]+[-  ]?[0-9]+$')
) COMMENT = 'Table to handle customer information';


-- 2. suppliers
CREATE TABLE bicycle_store.suppliers (
    supplierid INT PRIMARY KEY,
    companyname VARCHAR(100) NOT NULL,
    contactname VARCHAR(100) NOT NULL,
    contactemail VARCHAR(100) UNIQUE  NOT NULL,  -- unique and format expected xxx@yyy.zz
    contactphone VARCHAR(15),  -- format expected +xx-yyyyyyyyyy
            credit DECIMAL(10, 2) NOT NULL DEFAULT 0.00,
            CONSTRAINT  chk_contactemail_supp CHECK (contactemail REGEXP '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'),
            CONSTRAINT  chk_contactphone_supp CHECK (contactphone REGEXP '^\+[0-9]+[- ]?[0-9]+$')
) COMMENT = 'Table to handle supplier information';


-- 3. employees
CREATE TABLE bicycle_store.employees (
    employeeid INT PRIMARY KEY,
    firstname VARCHAR(50) NOT NULL,
    lastname VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,   -- unique and format expected xxx@yyy.zz
    phonenumber VARCHAR(15), -- format expected +xx-yyyyyyyyyy
    role VARCHAR(50) NOT NULL,
    hiredate DATE NOT NULL DEFAULT (CURRENT_DATE),
        CONSTRAINT chk_mail_empl CHECK (email REGEXP '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'),
        CONSTRAINT chk_phonenumber_empl CHECK (phonenumber REGEXP '^\+[0-9]+[- ]?[0-9]+$')
) COMMENT = 'Table to handle employee information';

-- 4. bicycles
CREATE TABLE bicycle_store.bicycles (
  bicycleid INT PRIMARY KEY,
  modelname VARCHAR(100) NOT NULL,
  brand VARCHAR(50) NOT NULL,
  type VARCHAR(50) NOT NULL,
  price DECIMAL(10, 2) NOT NULL DEFAULT 0.00,
  stockquantity INT NOT NULL DEFAULT 0,
        modelyear INT NOT NULL, -- format YYYY (1900-2050)
        specifications TEXT,
        photo BLOB,
        ingressdate DATE NOT NULL DEFAULT (CURRENT_DATE),
        supplierid INT,
        CONSTRAINT chk_year_bic CHECK (modelyear BETWEEN 1900 AND 2050),
        FOREIGN KEY (supplierid) REFERENCES bicycle_store.suppliers(supplierid)
) COMMENT = 'Table to store the bicycles available and their basic information';

-- 5. spareparts
CREATE TABLE bicycle_store.spareparts (
  sparepartid INT PRIMARY KEY,
  partname VARCHAR(100) NOT NULL,
  brand VARCHAR(50) NOT NULL,
  price DECIMAL(10, 2) NOT NULL DEFAULT 0.00,
    stockquantity INT NOT NULL DEFAULT 0,
            description TEXT,
            photo BLOB,
            ingressdate DATE NOT NULL DEFAULT (CURRENT_DATE),
            supplierid INT NOT NULL,
            FOREIGN KEY (supplierid) REFERENCES bicycle_store.suppliers(supplierid)
) COMMENT = 'Table to store the spare parts available and their basic information';


-- 6. appointments
CREATE TABLE bicycle_store.appointments (
    appointmentid INT PRIMARY KEY,
    appointmentdate DATETIME NOT NULL DEFAULT (CURRENT_DATE),
    purpose VARCHAR(100) NOT NULL,
    status VARCHAR(50) NOT NULL, -- 'Scheduled','Confirmed','Cancelled', 'Finished'
    customerid INT NOT NULL,
            employeeid INT NOT NULL,
            CONSTRAINT chk_status_app CHECK (status IN ('Scheduled','Confirmed','Cancelled', 'Finished')),
            FOREIGN KEY (customerid) REFERENCES bicycle_store.customers(customerid),
            FOREIGN KEY (employeeid) REFERENCES bicycle_store.employees(employeeid)
) COMMENT = 'Table to manage the appointment for repairs';


-- 7. orders
CREATE TABLE bicycle_store.orders (
    orderid INT PRIMARY KEY,
    orderdate DATETIME NOT NULL,
    totalamount DECIMAL(10, 2) NOT NULL DEFAULT 0.00,
    status VARCHAR(50) NOT NULL DEFAULT 'Pending', -- 'Cancelled','Payed','Pending'
            comments TEXT,
        paymentmethod VARCHAR(50),
        channel VARCHAR(50) NOT NULL DEFAULT 'Store', -- 'Store','App','WebPage', 'Partner'
    customerid INT NOT NULL,
        vendorid INT NOT NULL,
        CONSTRAINT chk_status_or CHECK (status IN ('Cancelled','Payed','Pending')),
        CONSTRAINT chk_channel_or CHECK (channel IN ('Store','App','WebPage', 'Partner')),
        FOREIGN KEY (customerid) REFERENCES bicycle_store.customers(customerid),
    FOREIGN KEY (vendorid) REFERENCES bicycle_store.employees(employeeid)
) COMMENT = 'Table to store the order information';


-- 8. orderdetails
CREATE TABLE bicycle_store.orderdetails (
    orderdetailid INT PRIMARY KEY,
    itemid INT NOT NULL,
    itemtype VARCHAR(20) NOT NULL, -- 'Bicycle' or 'SparePart' or 'Repair'
    quantity INT NOT NULL,
    unitprice DECIMAL(10, 2) NOT NULL DEFAULT 0.00,
    orderid INT NOT NULL,
        sparepartid INT,
        bicycleid INT,
        CONSTRAINT chk_status_od CHECK (itemtype IN ('Bicycle','SparePart','Repair')),
    FOREIGN KEY (orderid) REFERENCES bicycle_store.orders(orderid),
        FOREIGN KEY (sparepartid) REFERENCES bicycle_store.spareparts(sparepartid),
        FOREIGN KEY (bicycleid) REFERENCES bicycle_store.bicycles(bicycleid)
) COMMENT = 'Table to store the order details';


-- 9. repairs
CREATE TABLE bicycle_store.repairs (
  repairid INT PRIMARY KEY,
  problemdescription TEXT NOT NULL,
  status VARCHAR(50) NOT NULL, -- 'Cancelled','Ongoing','Finished'
  repaircost DECIMAL(10, 2) NOT NULL DEFAULT 0.00,
  egressdate DATE NOT NULL DEFAULT (CURRENT_DATE),
        ingressdate DATE NOT NULL DEFAULT (CURRENT_DATE),
        startdatetimerepair DATE DEFAULT (CURRENT_DATE),
        enddatetimerepair DATE DEFAULT (CURRENT_DATE), -- this value must be mayor of the startdatetimerepair
  bicycleid INT NOT NULL,
  customerid INT NOT NULL,
        CONSTRAINT chk_repairtime_rep CHECK (enddatetimerepair >= startdatetimerepair),
  CONSTRAINT chk_status_rep CHECK (status IN ('Cancelled','Ongoing','Finished')),
        FOREIGN KEY (bicycleid) REFERENCES bicycle_store.bicycles(bicycleid),
  FOREIGN KEY (customerid) REFERENCES bicycle_store.customers(customerid)
) COMMENT = 'Table to store the repair information';


-- 10. repairassignments
CREATE TABLE bicycle_store.repairassignments (
  assignmentid INT PRIMARY KEY,
  repairid INT NOT NULL,
  employeeid INT NOT NULL,
  FOREIGN KEY (repairid) REFERENCES bicycle_store.repairs(repairid),
  FOREIGN KEY (employeeid) REFERENCES bicycle_store.employees(employeeid)
) COMMENT = 'Table to store the details of the repair assignment with the technician';
```

Data Manipulation Statements (DML)

Now, we take the step to generate our data and insert everything on our tables. A useful tool to do this is mockaroo website. In this website we can create our columns and ask the system to generate random data around a topic. For example, we can create a column of emails, and the website has an option for creating this kind of data. Once the preparation on the website is done, the website generates the SQL file to import the data to populate our tables.

Examples for every insert are stated below.

```
-- Customers
INSERT INTO bicycle_store.customers (customerid, firstname, lastname, email,
phonenumber, address, membership, city, state, postalcode, country, creationdate)
VALUES (1, 'Rockey', 'Swindin', 'rswindin0@biglobe.ne.jp', '8703748771', 'Room 609',
'Basic', 'Niagara Falls', 'Ontario', 'T9H', 'Canada', '2023-12-14');
-- Suppliers
INSERT INTO bicycle_store.suppliers (supplierid, companyname, contactname, contactemail,
contactphone, credit) VALUES (1, 'Nitzsche, Breitenberg and Emmerich', 'Mathian Hasely',
'mhasely0@economist.com', '4771964640', 22410.49);
-- Bicycles
INSERT INTO bicycle_store.bicycles (bicycleid, modelname, brand, type, price,
stockquantity, modelyear, specifications, photo, ingressdate, supplierid) VALUES (1, 'Tarmac
SL7', 'Specialized', 'Road', 12000, 13, 2023, 'FACT 12r carbon frame, Shimano Dura-Ace Di2
groupset, Roval Rapide CLX wheels', null, '2024-01-13', 10);
-- Order Details
```

```
INSERT INTO bicycle_store.orderdetails (orderdetailid, itemid, itemtype, quantity, unitprice,
orderid, sparepartid, bicycleid) VALUES (1, 1, 'Bicycle', 2, 530.27, 1, 97, 24);
-- Repairs
INSERT INTO bicycle_store.repairs (repairid, problemdescription, status, repaircost,
egressdate, ingressdate, startdatetimerepair, enddatetimerepair, bicycleid, customerid)
VALUES (1, 'Full Tune-Up', 'Ongoing', '40.74', '2024-03-06', '2023-10-24', '2023-12-01',
'2024-01-25', 21, 570);
-- Employees
INSERT INTO bicycle_store.employees (employeeid, firstname, lastname, email,
phonenumber, role, hiredate) VALUES (1, 'Robb', 'Tooke', 'rtooke0@naver.com',
'6392576798', 'Manager', '2024-10-16');
-- Appointments
INSERT INTO bicycle_store.appointments (appointmentid, appointmentdate, purpose, status,
customerid, employeeid) VALUES (1, '2024-02-29', 'Plan Seasonal Tune-Up', 'Confirmed',
809, 3);

-- Repair Assignments
INSERT INTO bicycle_store.repairassignments (assignmentid, repairid, employeeid)
VALUES (1, 534, 1);
-- Orders
INSERT INTO bicycle_store.orders (orderid, orderdate, totalamount, status, comments,
paymentmethod, channel, customerid, vendorid) VALUES (1, '2024-09-04', 1268, 'Pending',
'tmckelvey0', 'Credit Card', 'Store', 34, 2);
```

```
-- Spare Parts
INSERT INTO bicycle_store.spareparts (sparepartid, partname, brand, price, stockquantity,
description, photo, ingressdate, supplierid) VALUES (1, 'Handlebar', 'Giant', 175, 27,
'Maecenas leo odio, condimentum id, luctus nec, molestie sed, justo. Suspendisse potenti.',
null, '2024-08-07', 2);
-- Order Details
INSERT INTO bicycle_store.orderdetails (orderdetailid, itemid, itemtype, quantity, unitprice,
orderid, sparepartid, bicycleid) VALUES (1, 1, 'Bicycle', 2, 530.27, 1, 97, 24);
```

For the **updates**, we though on problems that could had happen when populating such database. The cases are stated as follows:

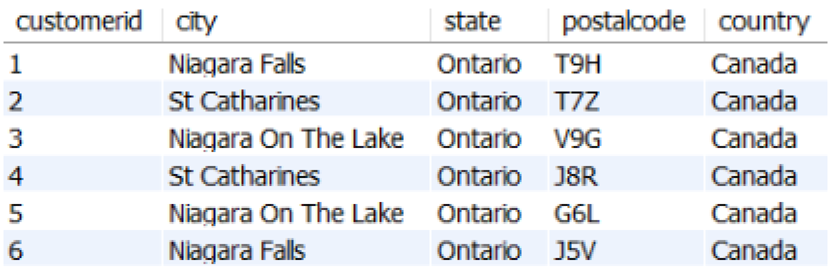
Case 1: The data register was loaded incorrectly, with these updates we are fixing with the correct information

```
UPDATE bicycle_store.customers
SET postalcode = 'L2E' WHERE city = 'Niagara Falls';
UPDATE bicycle_store.customers
SET postalcode = 'L2M' WHERE city = 'St Catharines';
UPDATE bicycle_store.customers
SET postalcode = 'L0S' WHERE city = 'Niagara On The Lake';
```

We can see the difference made by this update comparing Figure 1 and Figure 2.

Figure 1

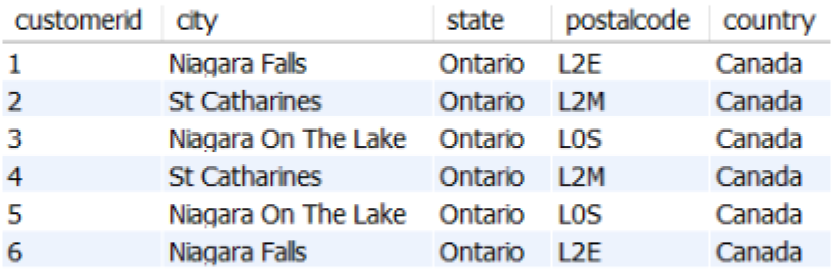
Customers: Before updating postal code



customerid	city	state	postalcode	country
1	Niagara Falls	Ontario	T9H	Canada
2	St Catharines	Ontario	T7Z	Canada
3	Niagara On The Lake	Ontario	V9G	Canada
4	St Catharines	Ontario	J8R	Canada
5	Niagara On The Lake	Ontario	G6L	Canada
6	Niagara Falls	Ontario	J5V	Canada

Figure 2

Customers: After updating postal code



customerid	city	state	postalcode	country
1	Niagara Falls	Ontario	L2E	Canada
2	St Catharines	Ontario	L2M	Canada
3	Niagara On The Lake	Ontario	L0S	Canada
4	St Catharines	Ontario	L2M	Canada
5	Niagara On The Lake	Ontario	L0S	Canada
6	Niagara Falls	Ontario	L2E	Canada

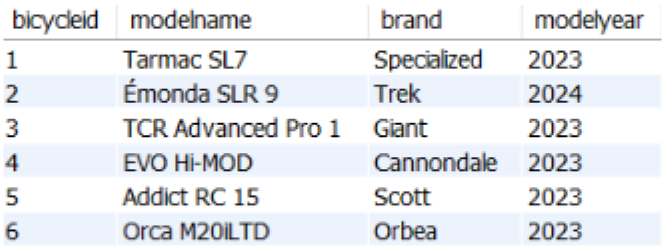
Case 2: The year loaded was the year out for sale, we changed this for the model year

```
UPDATE bicycle_store.bicycles

SET modelyear = modelyear+1;
```

We can see the difference made by this update comparing Figure 3 and Figure 4.

Figure 3

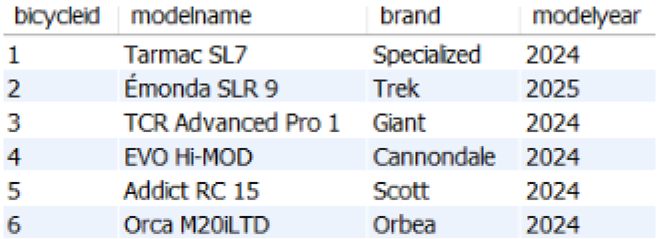
Bicycles: Before updating the model year



bicycleid	modelname	brand	modelyear
1	Tarmac SL7	Specialized	2023
2	Émonda SLR 9	Trek	2024
3	TCR Advanced Pro 1	Giant	2023
4	EVO Hi-MOD	Cannondale	2023
5	Addict RC 15	Scott	2023
6	Orca M20iLTD	Orbea	2023

Figure 4

Bicycles: After updating the model year



bicycleid	modelname	brand	modelyear
1	Tarmac SL7	Specialized	2024
2	Émonda SLR 9	Trek	2025
3	TCR Advanced Pro 1	Giant	2024
4	EVO Hi-MOD	Cannondale	2024
5	Addict RC 15	Scott	2024
6	Orca M20iLTD	Orbea	2024

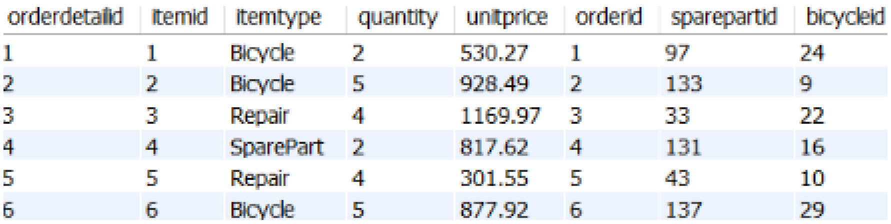
For the **alter**, we created a redundant field on the orderdetails table (itemid). The code to take respond to this problem is:

```
ALTER TABLE bicycle_store.orderdetails

DROP COLUMN itemid;
```

We can see the difference made by this update comparing Figure 5 and Figure 6.

Figure 5

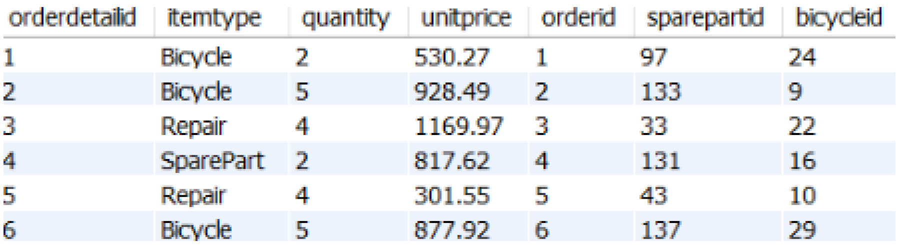
Orderdetails: Before altering the table by deleting itemid



orderdetailid	itemid	itemtype	quantity	unitprice	orderid	sparepartid	bicycleid
1	1	Bicycle	2	530.27	1	97	24
2	2	Bicycle	5	928.49	2	133	9
3	3	Repair	4	1169.97	3	33	22
4	4	SparePart	2	817.62	4	131	16
5	5	Repair	4	301.55	5	43	10
6	6	Bicycle	5	877.92	6	137	29

Figure 6

Orderdetails: After altering the table by deleting itemid



orderdetailid	itemtype	quantity	unitprice	orderid	sparepartid	bicycleid
1	Bicycle	2	530.27	1	97	24
2	Bicycle	5	928.49	2	133	9
3	Repair	4	1169.97	3	33	22
4	SparePart	2	817.62	4	131	16
5	Repair	4	301.55	5	43	10
6	Bicycle	5	877.92	6	137	29

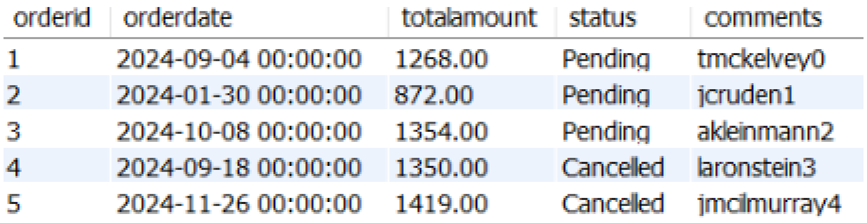
For **deleting** process, the registers in the table orders were loaded with the order date incomplete. Since is a problem that conflicts with different rows, we create the delete of the table orders_details and orders to load the data fixed.

```
DELETE

FROM bicycle_store.orders;

DELETE

FROM bicycle_store.orderdetails;
```

Once the data is deleted from the tables, we insert the new data. (files 12 and 13 of DML). We can see the results of this process in Figure 7 and Figure 8.

Figure 7

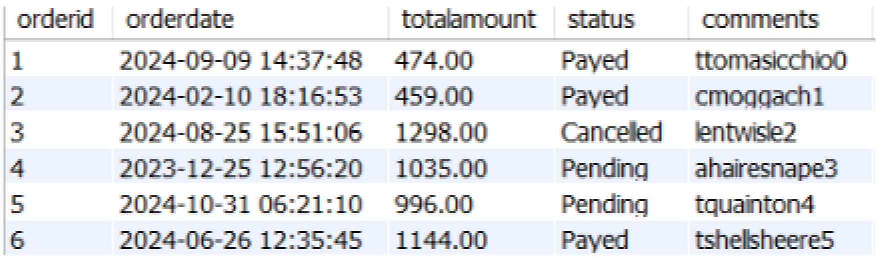
Orders: Before deleting the data to implement the new one



orderid	orderdate	totalamount	status	comments
1	2024-09-04 00:00:00	1268.00	Pending	tmckelvey0
2	2024-01-30 00:00:00	872.00	Pending	jcruden1
3	2024-10-08 00:00:00	1354.00	Pending	akleinmann2
4	2024-09-18 00:00:00	1350.00	Cancelled	laronstein3
5	2024-11-26 00:00:00	1419.00	Cancelled	jmcilmurray4

Figure 8

Orders: After deleting the data and implemented the new one



orderid	orderdate	totalamount	status	comments
1	2024-09-09 14:37:48	474.00	Payed	ttomasicchio0
2	2024-02-10 18:16:53	459.00	Payed	cmoggach1
3	2024-08-25 15:51:06	1298.00	Cancelled	lentwisle2
4	2023-12-25 12:56:20	1035.00	Pending	ahairesnape3
5	2024-10-31 06:21:10	996.00	Pending	tquainton4
6	2024-06-26 12:35:45	1144.00	Payed	tshellsheere5

Data Retrieval Statements (DQL)

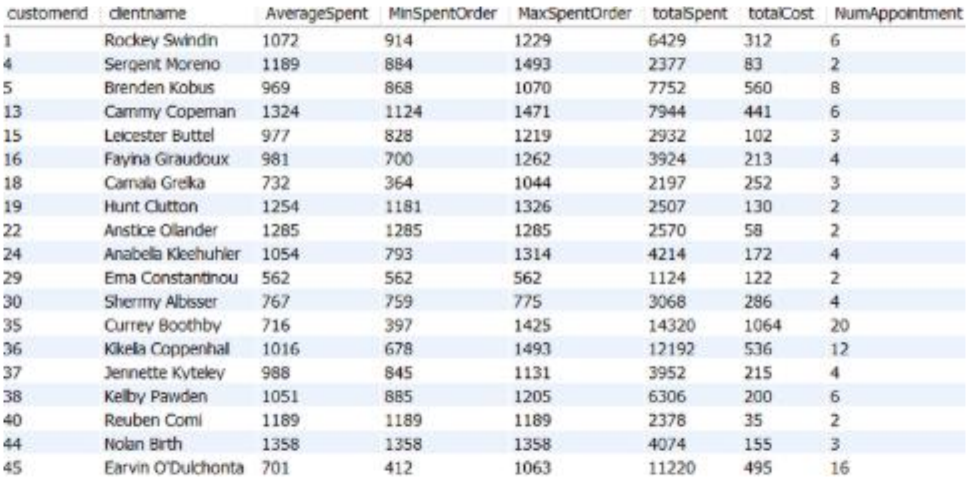
For this part, we though on giving view of descriptive statistics from all customers, the overall record of the shop and the transactions made by each customer. These insights have multiple purposes, such like financial projection, project evaluation, marketing analysis, etc.

We created a view with the average, min, max, total spend,cost of repairs, number of appointments by client, which can be seen in Figure 9.

```
CREATE VIEW bicycle_store.stattperclient  AS
SELECT
        cu.customerid,
   concat(cu.firstname,  " ", cu.lastname) AS clientname,
   ROUND(AVG(ord.totalamount),0) AS AverageSpent,
   ROUND(MIN(ord.totalamount),0) AS MinSpentOrder,
   ROUND(MAX(ord.totalamount),0) AS MaxSpentOrder,
   ROUND(SUM(ord.totalamount),0) AS totalSpent,
   ROUND(SUM(rep.repaircost),0) AS totalCost,
   COUNT(app.customerid) AS NumAppointment
FROM customers AS cu
JOIN orders AS ord ON cu.customerid = ord.customerid
JOIN repairs AS rep ON cu.customerid = rep.customerid
JOIN appointments AS app ON cu.customerid = app.customerid
GROUP BY cu.customerid
ORDER BY cu.customerid;
```

Figure 9

Stattperclient: Descriptive statistics for every customer



customerid	clientname	AverageSpent	MinSpentOrder	MaxSpentOrder	totalSpent	totalCost	NumAppointment
1	Rockey Swindin	1072	914	1229	6429	312	6
4	Sergent Moreno	1189	884	1493	2377	83	2
5	Brenden Kobus	969	868	1070	7752	560	8
13	Cammy Copeman	1324	1124	1471	7944	441	6
15	Leicester Buttel	977	828	1219	2932	102	3
16	Fayina Graudoux	981	700	1262	3924	213	4
18	Camala Grelka	732	364	1044	2197	252	3
19	Hunt Clutton	1254	1181	1326	2507	130	2
22	Anstice Ollander	1285	1285	1285	2570	58	2
24	Anabella Kleehuhler	1054	793	1314	4214	172	4
29	Ema Constantinou	562	562	562	1124	122	2
30	Shermy Albisser	767	759	775	3068	286	4
35	Currey Boothby	716	397	1425	14320	1064	20
36	Kikela Coppenhal	1016	678	1493	12192	536	12
37	Jennette Kyteley	988	845	1131	3952	215	4
38	Kelby Pawden	1051	885	1205	6306	200	6
40	Reuben Comi	1189	1189	1189	2378	35	2
44	Nolan Birth	1358	1358	1358	4074	155	3
45	Earvin O'Dulchonta	701	412	1063	11220	495	16

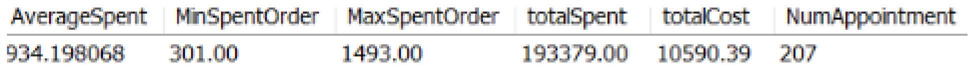
For the shop performance, we gathered the information in order to see the following information presented on Figure 10.

```
WITH summary AS (

SELECT

   AVG(ord.totalamount) OVER () AS AverageSpent,

   MIN(ord.totalamount)  OVER () AS MinSpentOrder,

   MAX(ord.totalamount)  OVER () AS MaxSpentOrder,

         SUM(ord.totalamount)  OVER () AS totalSpent,

   SUM(rep.repaircost)  OVER () AS totalCost,

   COUNT(app.customerid) OVER () AS NumAppointment

FROM customers AS cu

JOIN orders AS ord ON cu.customerid = ord.customerid

JOIN repairs AS rep ON cu.customerid = rep.customerid

JOIN appointments AS app ON cu.customerid = app.customerid)
```

```
SELECT *
FROM summary
LIMIT 1;
```

Figure 10

Summary: Summary of the shop's performance



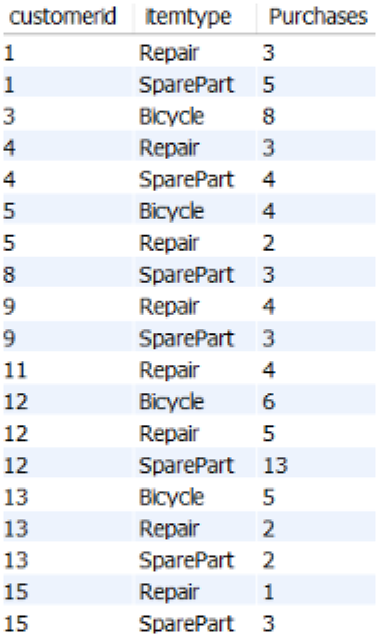
AverageSpent	MinSpentOrder	MaxSpentOrder	totalSpent	totalCost	NumAppointment
934.198068	301.00	1493.00	193379.00	10590.39	207

Lastly, we check on the summary of the customer's purchases. This data can be the initial material to conduct a machine learning process to see what type of service/product is the one with more transactions. The query's result is presented on Figure 11.

```
SELECT
o.customerid,
odt.itemtype,
SUM(odt.quantity) AS Purchases
FROM orderdetails odt
JOIN orders o ON o.orderid=odt.orderdetailid
GROUP BY o.customerid, odt.itemtype
ORDER BY o.customerid;
```

Figure 10

Summary: Summary of the shop's performance



customerid	itemtype	Purchases
1	Repair	3
1	SparePart	5
3	Bicycle	8
4	Repair	3
4	SparePart	4
5	Bicycle	4
5	Repair	2
8	SparePart	3
9	Repair	4
9	SparePart	3
11	Repair	4
12	Bicycle	6
12	Repair	5
12	SparePart	13
13	Bicycle	5
13	Repair	2
13	SparePart	2
15	Repair	1
15	SparePart	3